ECE482 — Introduction to Operating Systems

Homework 2

Manuel — JI (Fall 2024)

Non-programming exercises:

- Write in a neat and legible handwriting
- Clearly explain the reasoning process
- Write in a complete style (subject, verb and object)

Progamming exercises:

- Write a single README file per homework
- Push to git and create a release with tag h2

ECE4821: submit together with ECE4820

ECE4820 Exercises

Ex. 1 — *Multiprogramming*

A few years ago when computers featured less RAM it is was common to increase it in order to enhance CPU performance. In order to better understand the link between the two we now create a simple model for multiprogramming. We assume all the processes to be similar and spending the same fraction p of their time waiting for Input/Output (I/O) to complete.

1. What is the probability for n processes to be waiting at the same time, then express the CPU utilisation as a function of n?
2. Sketch the curve representing the CPU utilisation as a function of the number of processes for the following values of p: 25%, 60% and 90%.
3. A certain old computer has 256 MB of RAM, once loaded a light operating system uses 96 MB of RAM. Several programs are launched each of them using 48 MB.
   a) How many processes can be store simultaneously in memory?
   b) Assuming an average of 90% I/O waiting time what is the CPU utilisation?
   c) What is the effect of adding 256 MB, 512 MB and 1024 MB of RAM. Argue on which amount would be the most beneficial and would be worth the investment.

Ex. 2 — *Understanding system calls*

1. Briefly introduce `strace` and `ltrace`. Explain they could be helpful along the semester.
2. What are the `manpages` sections for system calls and library calls?
3. System calls.
   a) What are the main types of system calls?
   b) Run `strace` on the `ls` command and classify all the listed system calls.
   c) Select a line of your choice in `strace` output and explain its meaning.
   d) How to attach `strace` to a running process? Describe a scenario where this could be handy.

ECE4821 Exercise

Ex. 3 — *A simple system call*

1. Kernel printing.

a) What is the counterpart of `printf()` when working inside the Linux kernel?

b) Write the body a simple "kernel function" displaying "Mum is proud of you!"

2. Headers and function name.

   a) What header files should be included when adding a new system call?

   b) How are the macros `SYSCALL_DEFINE0` and `SYSCALL_DEFINEx` working?

   c) Use the `SYSCALL_DEFINE0` to define the new system call `encouragement`. Name your file `encouragement.c` and save it in a folder `encouragement` at the root of the kernel source code.

3. Makefile and Kbuild.

   a) Jump to the end of `Kbuild` file and following the same patterns as on the last few lines, add your `encouragement` directory there.

   b) Check the Makefiles for the directories listed at the end of `Kbuild` file and create a Makefile for your `encouragement` directory.

4. System call registration.

   a) Add your new system call to `syscalls.h`.

   Hint: there are more than one such file, make sure to edit the correct one!

   b) Add your new system call to `syscall_64.tbl`.

   Hint: what architecture is your Linux running on?

5. Recompile the kernel and reboot (do not forget to update the bootloader).

6. Write a simple user space program to demonstrate the well functioning of your system call.